Jathushan Karthigesar

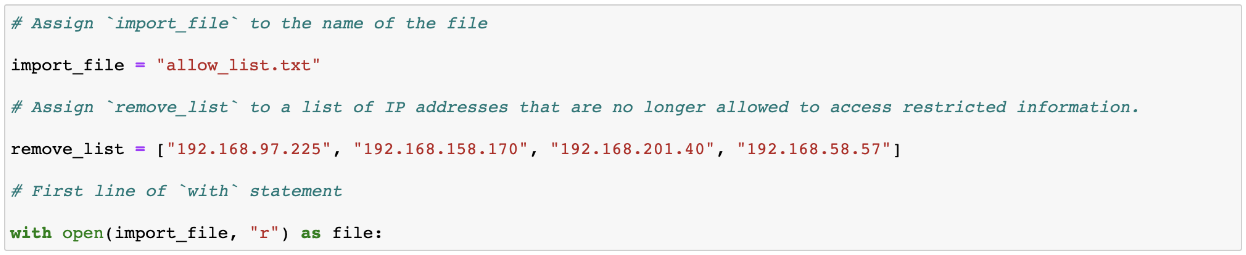
Algorithm for file updates in Python

Project description

I am required to update a file that identifies the employees who can access restricted content. I created an algorithm that uses Python code to check whether the allow list contains any IP addresses identified on the remove list. If so, I removed those IP addresses from the file containing the allow list.

Open the file that contains the allow list

I need with statement to handle the file. I need open() function to open the file. This function will take to parameter (1st parameter is the file name and second parameter will be the “r” command to specify tht we are opening the file for reading purpose.) Then we need the as keyword to tell python to use the variable file to store the file while you work with it inside the with statement. Finally, we must add a colon at the end the with statement header for syntax purpose.



```python
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# First line of `with` statement

with open(import_file, "r") as file:
```

Read the file contents

Now that the allow list file contents are saved in the file variable and we are working within the with statement, we will need the .read() method to be appended after the file variable; file.read(). This method will convert the contents of the allow list file into a string so that you can read them. Then we need store this string in a variable called ip_addresses; ip_addresses = file.read(). This line of code should be indented since it is being done within the with statement.

```python
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Display `ip_addresses`

print(ip_addresses)
```

Convert the string into a list

We need the .split() method to convert the ip_addresses string into a list. So since ip_addresses variable contains the string we want to convert, we append .split() method to it; ip_addresses.split(). Then we assign the resulting list to ip_addresses variable so we can reuse the list simply by calling ip_addresses. Since this done outside the with statement, this line of code does not need to be indented.

```python
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Display `ip_addresses`

print(ip_addresses)
```

Iterate through the remove list

We need the for keyword to indicate the beginning of the for loop. We need element loop variable to temporarily store the elements in ip_addresses. We need in keyword to specify where we want to loop (in this case, ip_addresses list). Final we must add a colon at the end of the for loop header for syntax purpose.

```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

    # Display `element` in every iteration

    print(element)
```

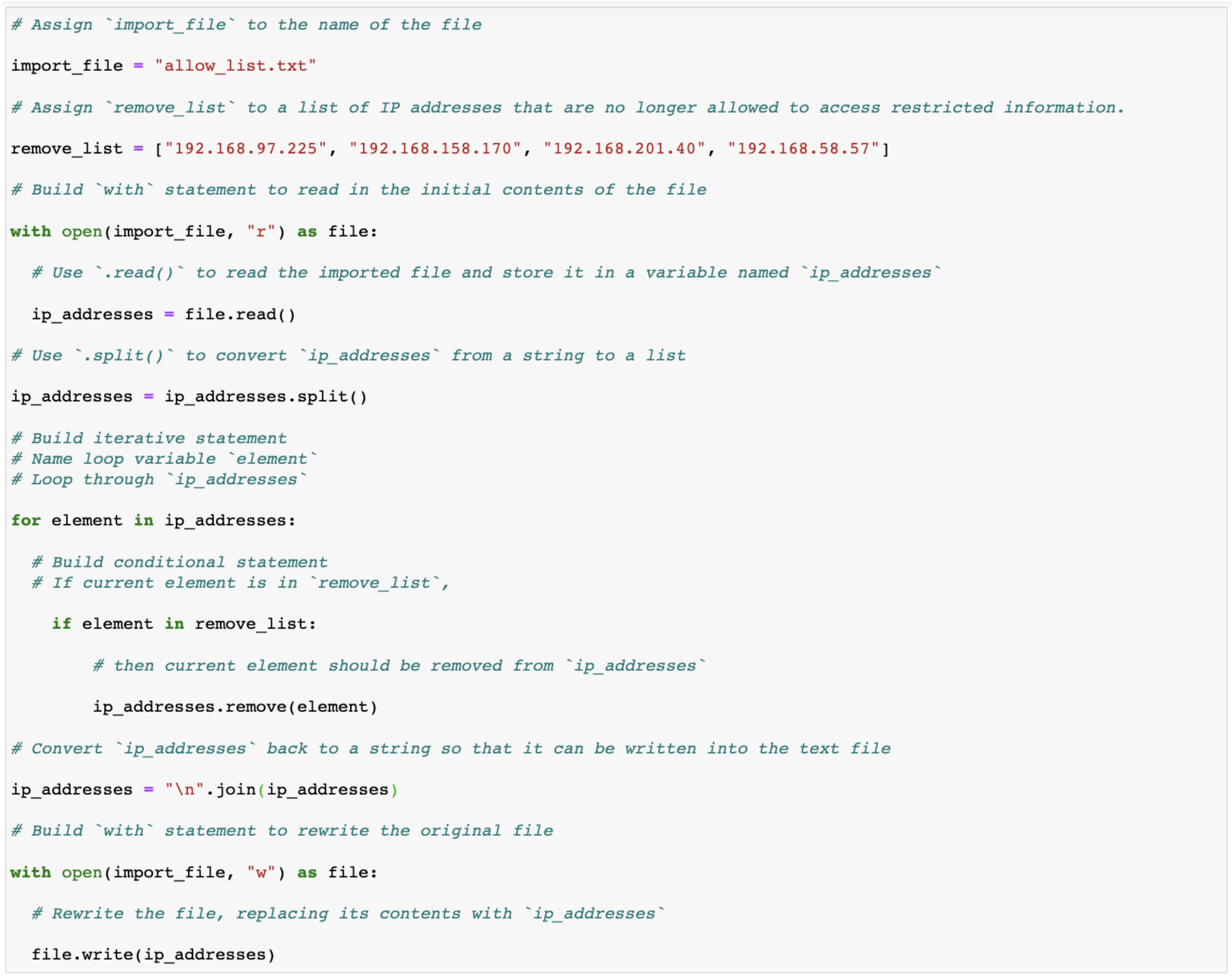
Remove IP addresses that are on the remove list

Inside the for loop, we have to make an indented line that is a conditional statement header that checks if the loop variable element is part of the remove_list list: “ if element in remove_list:”. We have to add a colon at the end of the header of the conditional statement for syntax purposes. Then, within that conditional, we indent a new line where we apply the .remove() method to the ip_addresses list and remove the IP addresses identified in the loop variable element if that IP address is in the remove_list: ip_addresses.remove(element). Applying the .remove() method in this way is possible because there are no duplicates in the ip_addresses list.

```
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)

# Display `ip_addresses`

print(ip_addresses)
```

Update the file with the revised list of IP addresses

We must convert the ip_addresses list back into a string using the .join() method. The .join method takes the iterable it wants to convert as input. We have to apply .join() to the string "\n" in order to separate the elements in the string by placing them on a new line. Thus the code line will be written as "\n".join(ip_addresses). Then we need to store this string in a variable called ip_addresses so that we can call ip_addresses whenever we want to use the string; ip_addresses = "\n".join(ip_addresses).



```python
# Assign `import_file` to the name of the file

import_file = "allow_list.txt"

# Assign `remove_list` to a list of IP addresses that are no longer allowed to access restricted information.

remove_list = ["192.168.97.225", "192.168.158.170", "192.168.201.40", "192.168.58.57"]

# Build `with` statement to read in the initial contents of the file

with open(import_file, "r") as file:

  # Use `.read()` to read the imported file and store it in a variable named `ip_addresses`

  ip_addresses = file.read()

# Use `.split()` to convert `ip_addresses` from a string to a list

ip_addresses = ip_addresses.split()

# Build iterative statement
# Name loop variable `element`
# Loop through `ip_addresses`

for element in ip_addresses:

  # Build conditional statement
  # If current element is in `remove_list`,

    if element in remove_list:

        # then current element should be removed from `ip_addresses`

        ip_addresses.remove(element)

# Convert `ip_addresses` back to a string so that it can be written into the text file

ip_addresses = "\n".join(ip_addresses)

# Build `with` statement to rewrite the original file

with open(import_file, "w") as file:

  # Rewrite the file, replacing its contents with `ip_addresses`

  file.write(ip_addresses)
```

Summary

There is an allow list for IP addresses permitted to sign into the restricted subnetwork. There's also a remove list that identifies which employees you must remove from this allow list. Thus the algorithm I wrote converts the allow list file contents into a string, and then to a list. Then the algorithm removes IP addresses indicated in the remove list from the “allow list file” list. Then the algorithm converts the updated list back to a string so that it can be used to replace the allow list file contents through the .write() method.